International Journal of Control and Automation
Vol. 11, No. 4 (2018), pp.63-74
http//dx.doi.org/10.14257/ijca.2018.11.4.06

Design and Implementation of Actuator Middleware Based on ID/IP Using RESTful API for Controlling IoT Appliances

Sehrish Malik[1], HeeDong Park[2] and Do-Hyeun Kim[3*]

[1,3]Dep. of Computer Engineering, Jeju National University, Republic of Korea
[2]School of IT Convergence, Korea Nazarene University, Republic of Korea
[1]serrym29@gmail.com, [2]hdpark@kornu.ac.kr, [3]kimdh@jejunu.ac.kr*

Abstract

The deployment of sensors and actuators at an accelerated rate can be observed in our surroundings with each passing day. Data from sensors is processed to get useful information and to derive decisions in order to control the actuators. This smart control of actuators is enabled with the deployment of intermediary layer, actuator middleware, which manages the actuator control functionality. In this paper, we propose an actuator middleware model based on ID/IP for multi actuator networks, and identification mapping mechanism between ID and IP. Also, we design and implement the proposed actuator middleware within the RESTful actuator control, and test with the device emulators and actuator module on Edison board. Our developed system can be used as a testbed for multi-actuator networks.

Keywords: *IoT (Internet of Things); actuator platform; actuator middleware; actuator control; RESTful API*

1. Introduction

The vision of the Internet of Things (IoT) is to connect the devices all over the Internet, enabling them to share and exchange data in order to provide useful services to people. A primary goal of interconnecting these devices and collecting/processing data from them is to create situation awareness and enable applications, machines, and human users to better understand their environments. These devices generate huge amount of data to be acquired by many services and application in areas such as smart homes, smart grids, healthcare, and environmental monitoring [1].

Infusion of intelligence into everyday things inside a home, an office or any other given environment makes it a smart environment. In a smart environment, things are capable of making decisions based on results gathered from data collected from surroundings. The process of deriving intelligent decisions from the sensed data from an indoor environment and controlling the actuators is called smart control [2]. There can be many different ways to implement and manage smart control. To simplify this task, the application and actuator communication logic can be separated by introducing an interface layer in between which is called as actuator middleware layer. In this paper we propose an actuator middleware for managing the smart control of actuator network based on ID/IP mappings.

ID/IP mappings is to maintain the actuator address relation information received from actuator middleware and updating indoor appliances the operational state information of the actuators at the actuator platform. Smart control also contains other components as control message for actuator, control result which is the response message that results in the execution or rejection of control command, updating mapping table that is to update

Received (January 3, 2018), Review Result (March 9, 2018), Accepted (March 12, 2018)
* Corresponding Author

ISSN: 2005-4297 IJCA

actuator address relation information received from actuator middleware and updating the operational state information of the actuators.

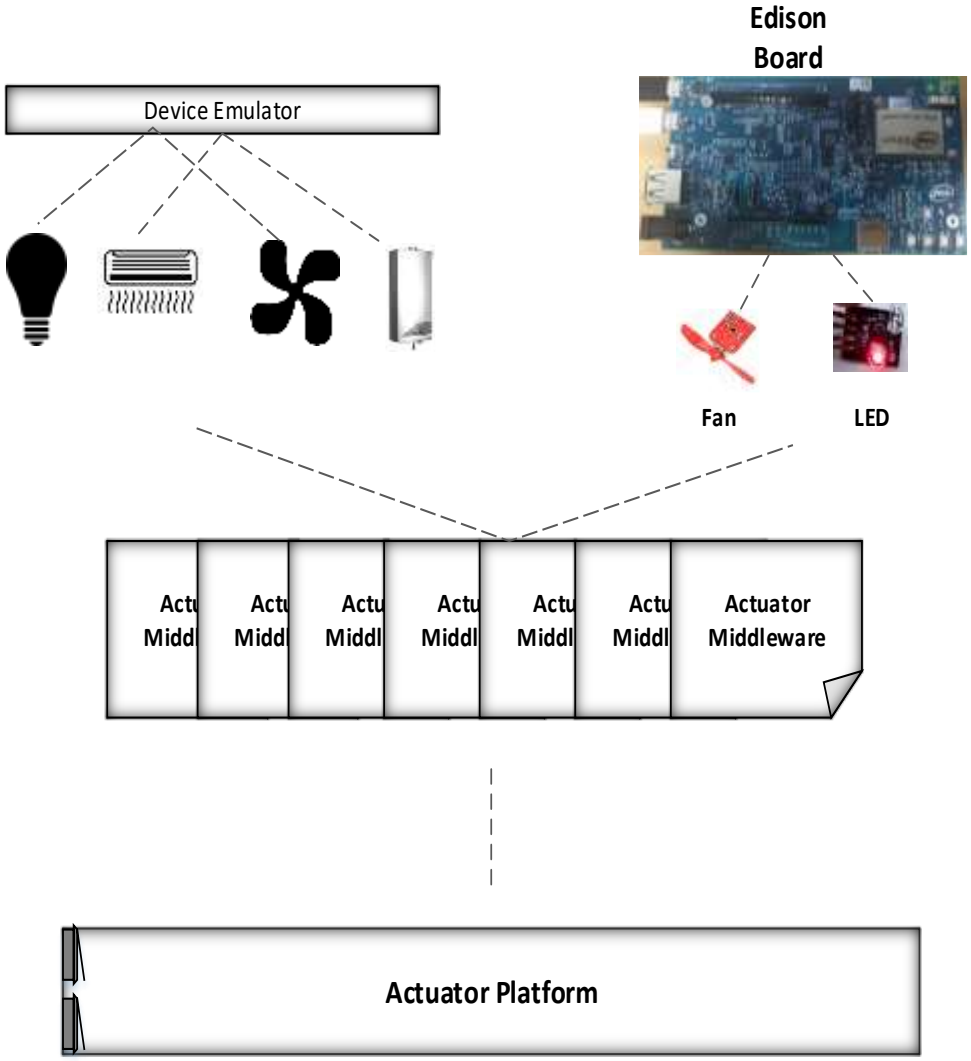

Figure 1. Conceptual Configuration of Actuator Middleware

This work is an extended version of the previous work published in CRTT 2017 [3]. Figure 1 shows the conceptual configuration of the presented work. We have developed an actuator web control system and testbed, which provides service for remotely controlling the indoor appliances. Thus client generates command through actuator platform, actuator middleware executes the command on the actuator and the client receives the response message back through the actuator platform. The actuator middleware collects actuator state information and sends control commands to the actuators. At one end of actuator middleware will be a RESTful actuator platform, to access and store the actuator resources. RESTful is an architectural style for web services [4]. On the other end of actuator middleware we'll have deployment for our actuator network. We have two implementations for the actuator network demonstration, one is device emulators and the other is Edison board. Device emulators virtually show the working of actuators, we have developed emulators for the four actuators such as fan, light, boiler and air-conditioner. The communication medium between both the actuator middleware and actuator, and the actuator middleware and actuator platform is TCP port.

We also tested the middleware using fan and light modules on Edison board with Linux server on the board and using establishing Wi-Fi connection between the actuator middleware and Edison server.

The rest of the paper is organized as follows, Section 2 presents related work. Section 3 illustrates the actuator middleware design. In Section 4, we present the environment and implementation details along with results while Section 5 concludes the paper.

2. Related Work

The presented work in [5], provides a framework, Sentire, to build extensible application middleware that uses sensor and actuator networks. The proposed SANET system has three layers as application layer, middleware layer and SANET layer. Application layer has number of application connected to Sentire middleware layer. The middleware has resource optimization, QoI management, query management and data processing. On the other end of middleware is SANET layer which consists of sensors and actuator networks. A case study is presented in order to demonstrate the working of Sentire framework.

The work in [6], presents a Java middleware, SENSACT API package for mobile pervasive augmented reality games. It connects sensor and actuator networks via Bluetooth network; creating, managing and communicating the main components of the given personal area. Simulated of multiple sensors and actuators is done using Java Wireless Toolkit which shows that the proposed middleware solution is good enough for the small sensors and actuators networks that have a center coordinating device similar to mobile and pervasive augmented reality scenarios.

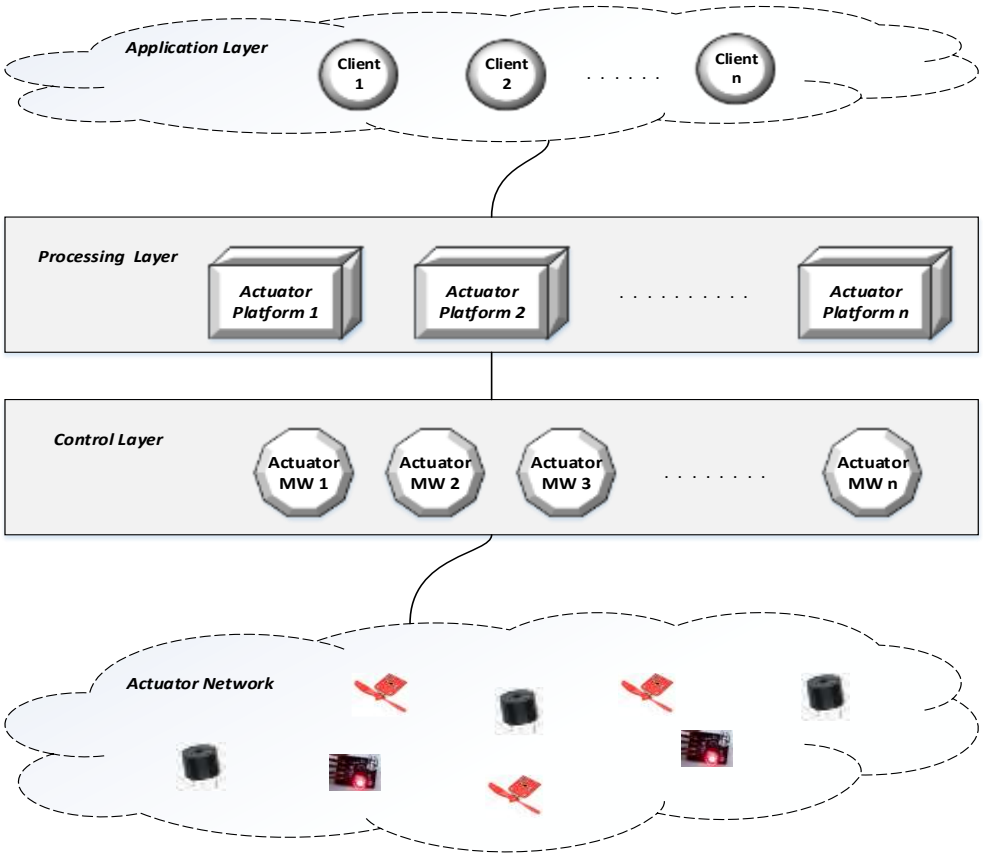

Figure 2. IoT Architecture Based on Actuator Middleware

The work in [7], proposes a middleware for sensor-actuator networks which cooperate with the aerial vehicles. Aerial vehicles are the autonomous helicopters. The proposed features in the system are fire detection, confirmation, and localization, monitoring of the given environment and node deployment. The middleware consists of three main components as routing engine, filtering engine and gateway management engine. The system tends to be helpful in disaster management where helicopters can monitor an area, can data from sensors deployed and forward data via middleware for efficient decision making.

The work in [8], presents a service oriented model driven approach to develop systems to increase the reusability of the implemented code. They aim to develop services in a way so that any sub-components can be combined to make an application when required. The interactions between the components are done via middleware, which is tailored for each application. The middleware also aims to provide optimal size and run-time behavior for the system. Middleware is tailored using a domain specific, template based code generator. The work provides a way for platform specialists, domain experts and end users to focus on building their part, making things easier, efficient and reusable.

3. Design of Actuator Middleware

In this section, we present the design of our proposed approach for actuator middleware. Figure 2 shows the IoT architecture based on actuator middleware included layers for actuator control. There are main layers as application layer, processing layer, control layer and actuator network layer. Application layer connects with the actuator platform and passes on the data and commands to and from the clients, processing layer has actuator platforms connected to the actuator network layer with the help of actuator middleware which acts as an intermediary between top layer and actuator network.

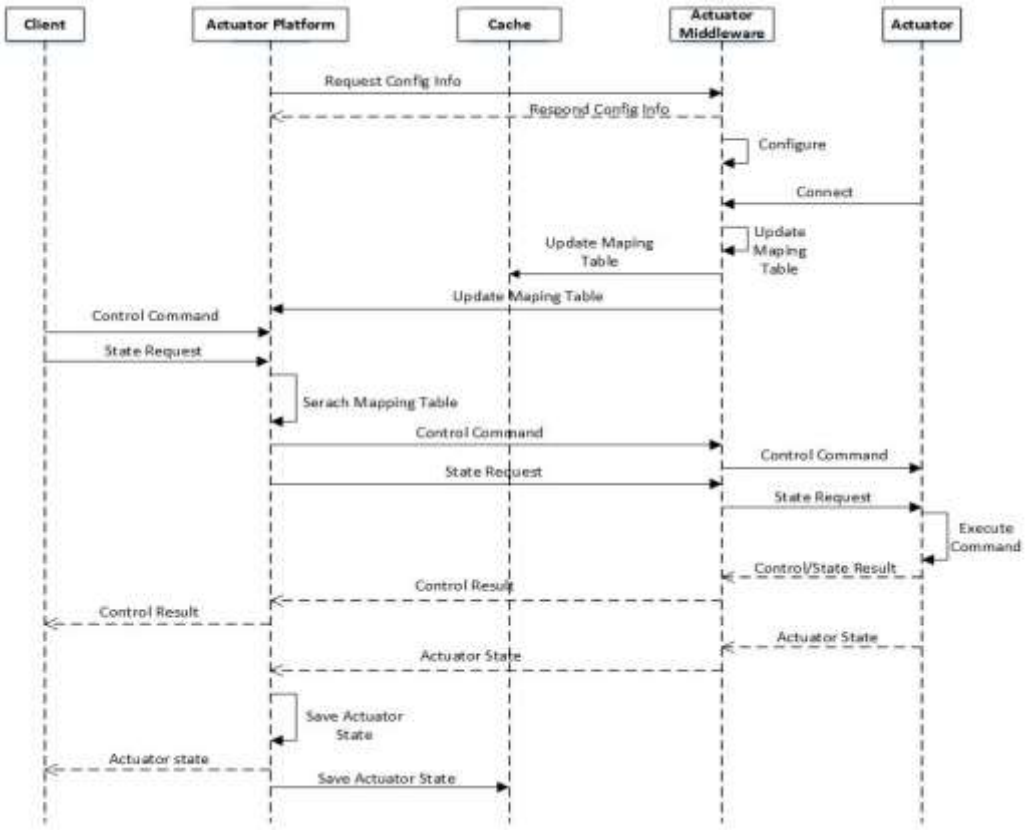

Figure 3. Sequence Diagram of Actuator, Middleware, and Platform

System's sequence diagram is shown in Figure 3. First the configuration between the actuator platform and the middleware is done. Then actuator connects to the service provider via middleware; it sends connect request to middleware; middleware updates the mapping table when actuator is connected and forwards the update mapping table request to the actuator platform. Middleware also updates the mapping in the cache. Client sends control command or state view request to the actuator platform. Actuator platform searches the involved actuator in the mapping table and forwards the command/request to the responsible middleware. Middleware forwards the command/request to the actuator. Command is executed at actuator and response is sent to the middleware. Middleware forwards the command response to the actuator platform which then forwards the response to the client. Actuator then sends the actuator status to the middleware which then forwards it to the client via actuator platform. Actuator platform keeps updating the mapping table every time actuator is connected or disconnected or any other changes to be updated.

Figure 4 depicts the configurations of the actuator middleware. Actuator middleware acts as an intermediary between actuator and the actuator platform. It receives messages from the actuator platform and actuator, parses the messages and sends back the response after processing. Middleware configuration management involves the creation of middleware ID, IP address and service access right information and management. Also enables the server connection of the actuator via middleware. It maps the middleware IP and actuator ID and updates the mappings when needed. In RESTful, cache is accessed for any data retrieval, if not found then DB is accessed.

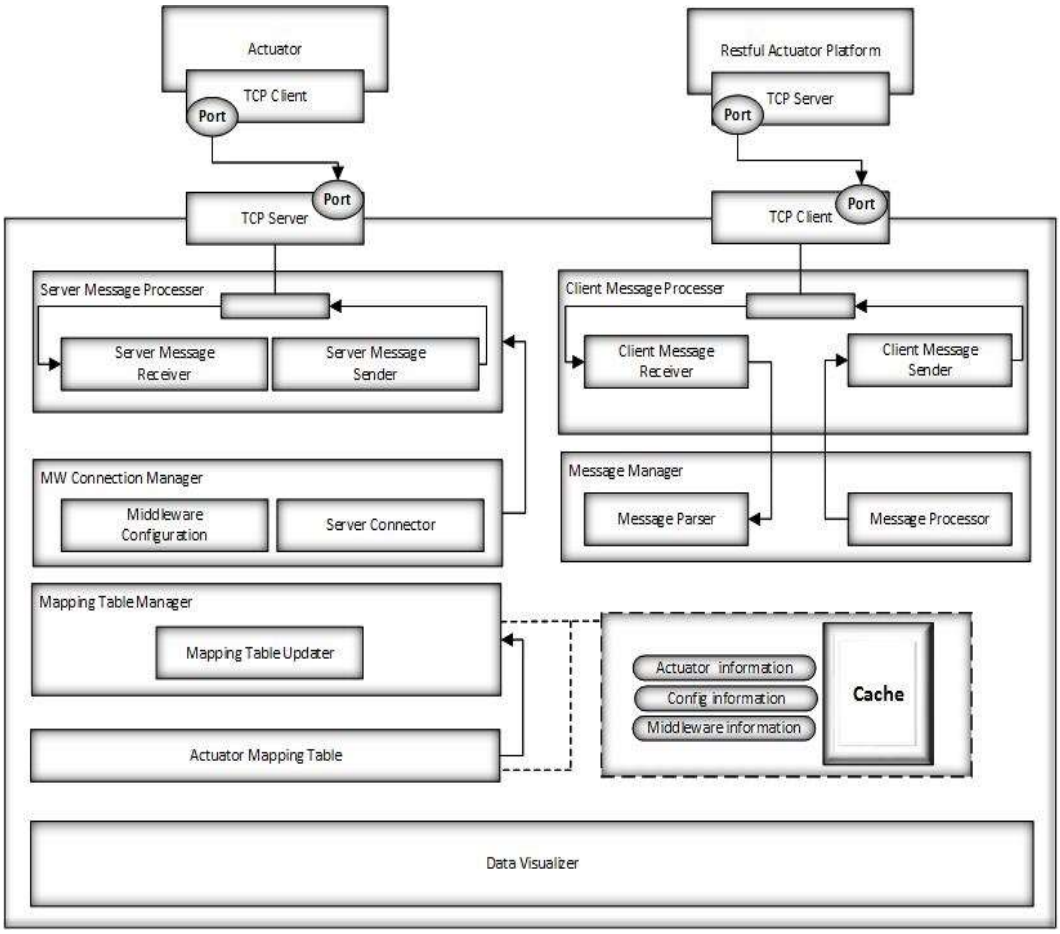

Figure 4. Actuator Middleware Configurations

Figure 5 shows the mapping table creation and management for the actuator IDs and middleware IPs for connectivity of actuator, middleware, and platform. First the actuator sends a connection request to the middleware, after receiving request, middleware maps an actuator platform ID to the actuator ID and forwards the actuator connection information to the mapped actuator platform. Actuator platform receives the actuator information, compares the actuator status and mapping information in its own mapping table and updates the middleware IP and actuator ID mappings.

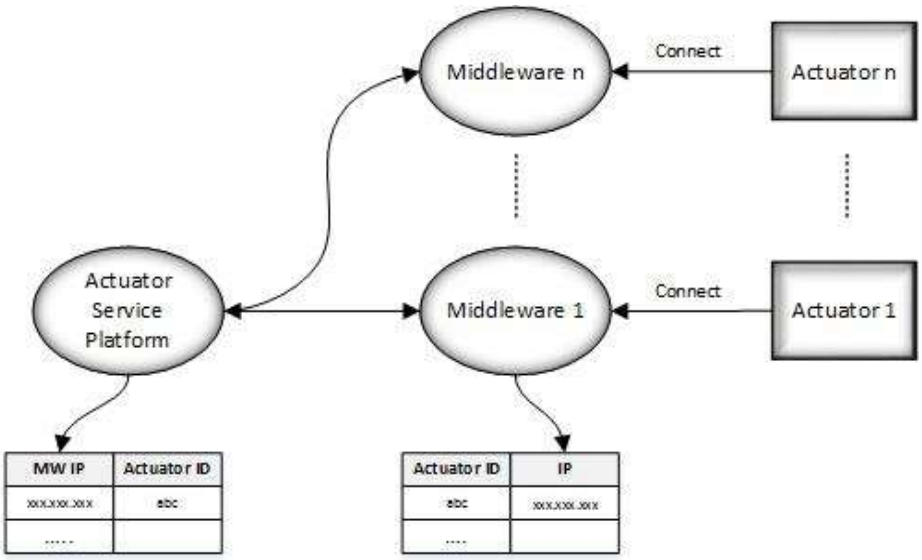

Figure 5. ID/IP Mapping Table Management for Connectivity of Actuator, Middleware, and Platform

Figure 6 shows the connection sequence between actuator middleware and actuator module mounted on Edison board. We have developed a CoAP.Net server on Intel Edison, to which the actuator middleware connect as client to send and receive control messages for actuator.

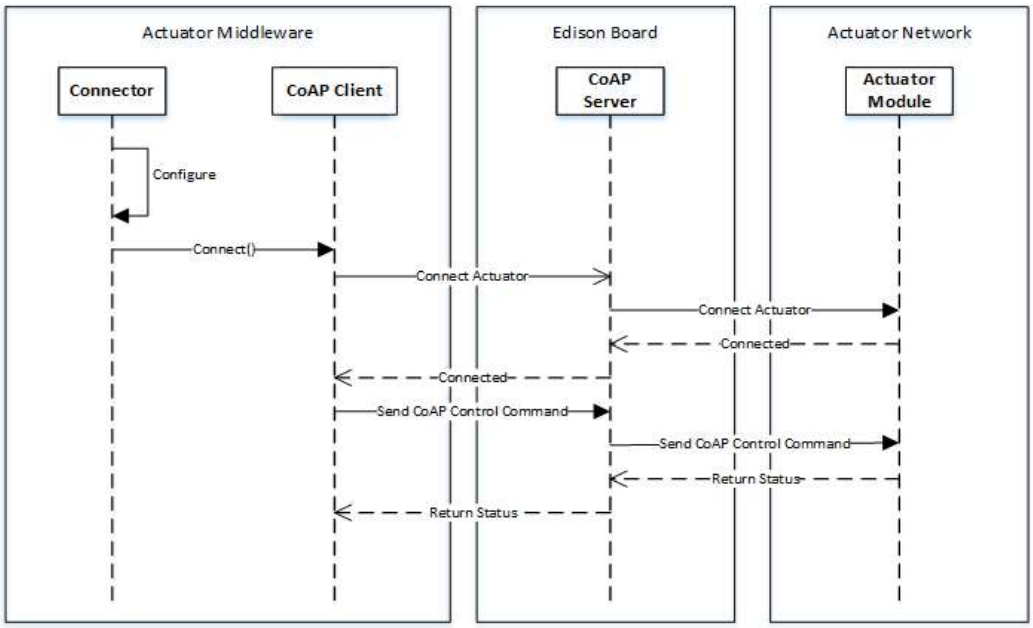

Figure 6. Connecting Actuator Middleware and Edison Board

4. Implementation of Actuator Middleware

The implementation environment for our system is described in table 1. For actuator platform and middleware implementation, we have used Windows 10 pro (x64) as the operating system, Visual Studio 2015 as development tool and Microsoft SQL Server as database management system. For actuator network connection testing, actuator emulators are built on Microsoft visual studio and the testing connection of actuator with Edison board is also done using fan and LED modules on Intel Edison with Kit for Arduino. Edison board will have micro CoAP controllers, Lib CoAP of C version, and Yocto and Linux environment.

Table 1. Implementation Environment

Components	Version
Operating System	Microsoft Windows 10 pro (x64)
Microsoft Visual Studio	2015
Microsoft SQL Server Management Studio	2016
Intel System Studio IoT Edition	2017
Putty.exe	2014
Yocto Linux	2014
Intel Edison with Kit for Arduino	2014

Actuator platform is built on RESTful API. In RESTful, we need to define URIs for performing tasks between the modules. Figure 7 shows a few of the defined URIs in content service of actuator platform.

```
[OperationContract]
[WebInvoke(UriTemplate = "/SendMessageToActuator?msg={msg}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
bool SendMessageToActuator(string msg);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuator?nActuatorID={nActuatorID}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
Actuator GetActuator(string nActuatorID);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorList?KeyWord={KeyWord}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
List<string> GetActuatorList(string KeyWord);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorModel?mModelCode={mModelCode}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
ActuatorModel GetActuatorModel(long mModelCode);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorType?mTypeCode={mTypeCode}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
TypeInformation GetActuatorType(long mTypeCode);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorAttributes?mModelCode={mModelCode}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
List<ActuatorAttribute> GetActuatorAttributes(long mModelCode);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorAccessState?nActuatorID={nActuatorID}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
bool GetActuatorAccessState(string nActuatorID);

[OperationContract]
[WebInvoke(UriTemplate = "/getActuatorState?ActuatorID={ActuatorID}", Method = "GET", ResponseFormat = WebMessageFormat.Json)]
List<StateData> GetActuatorState(string ActuatorID);
```

Figure 7. Defined URI in Actuator Platform

Figure 8 shows the actuator middleware execution screen. Connect Actuator connects the actuator to the service provider via middleware. Once the actuator connection is established with middleware, its ID and IP are shown in the list. History presents the timeline of all the actuator events. It shows what time and what function was performed on the actuator. Actuator ID and Actuator IP mappings show which actuator is connected to which actuator platform.

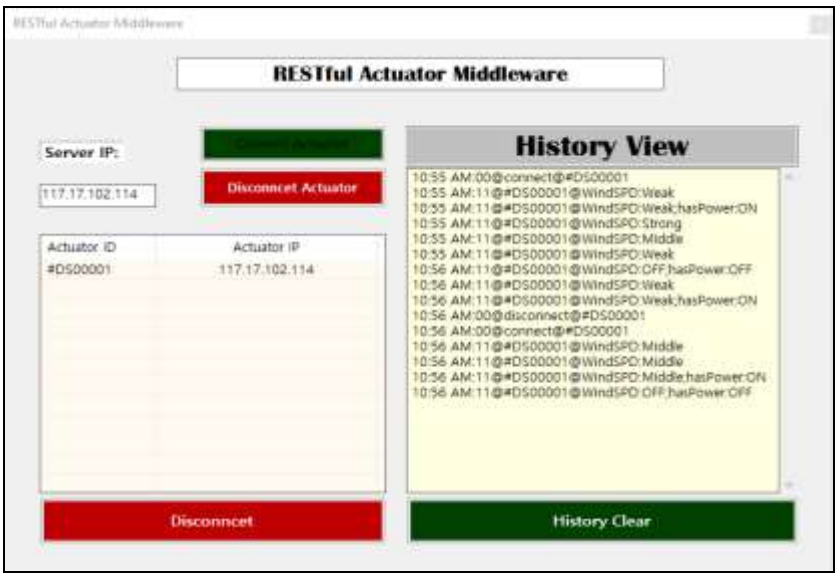

Figure 8. Implementation of Actuator Middleware.

Smart home appliances can be equipped with the sensor chips and a user can connect remotely to them using network/internet. Here, an emulator is implemented to imitate real-world appliances. Fan emulator, light emulator, boiler emulator and air conditioning emulator can be called, by using actuator object generation tool as shown in Figure 9.

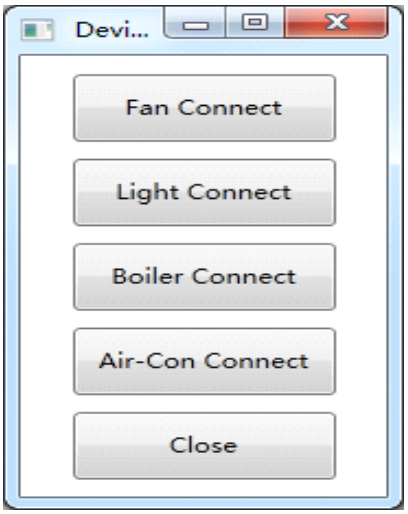

Figure 9. Device Emulator

Figure 10 shows the fan emulator screen. It offers functions like airflow adjustment and power control. Wind Level State shows fan current wind speed rating. Wind Level Control can be adjusted to three levels of wind speed i.e. Strong, Middle and Weak. Power State can adjust power ON/OFF.

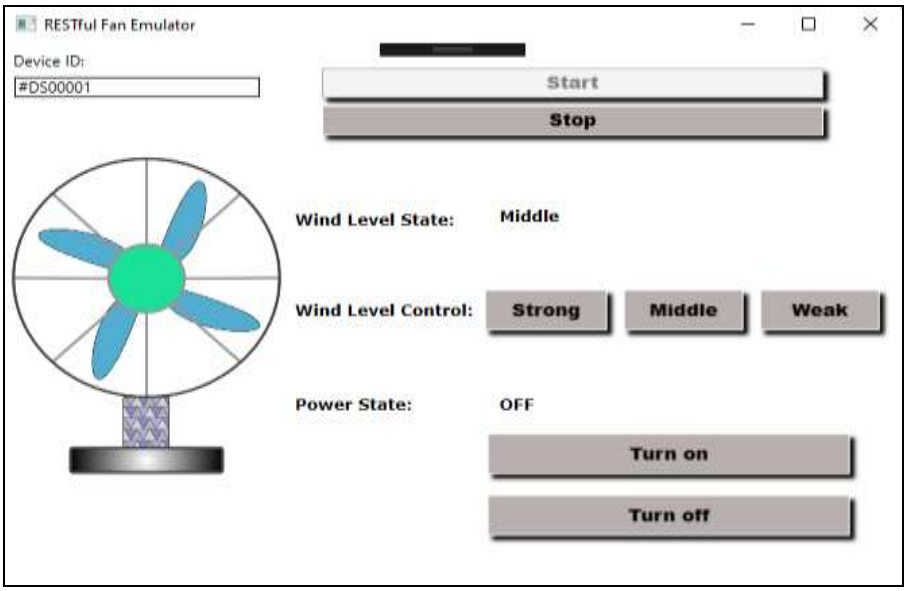

Figure 10. Fan Emulator

Also, we implement the actuator connection from Edison board to middleware. In this case, actuator middleware exchange messages with CoAP server, for IoT Device and Actuators. Then we use the Edison board for IoT Device, and fan and light modules actuators. They connected with Edison. Figure 11 shows the Edison board, fan and light modules.

(a) Edison board (b) Fan and LED

Figure 11. IoT Device and Actuators

5. Conclusion

This work proposed an actuator middleware design for actuator control platform. With the rapid increase in sensors and actuators applications; this research area is becoming of great importance. Our approach provides a flexible multi-middleware for actuator networks. The IP, ID mappings, make it very convenient and easier to keep track of devices connected from different sources. Our proposed middleware can be embedded with different actuator networks. It can act as a client for any new sub-module of actuator, with few modifications, making the actuator platform widely workable, as it will enable the clients to control and command the actuators from different actuator network implementations.

We have proposed detailed architecture of middleware for an indoor actuator network and also have tested its implementation with both virtual and physical devices. Implemented actuator system has an actuator platform based on RESTful API, we have also built similar platform with SOAP API. In future, we plan to take out detail comparison analysis between RESTful and SOAP platforms with different actuator network sources.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2017R1D1A3B06035024), and this research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Promotion). Any correspondence related to this paper should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr.

References

[1] https://en.wikipedia.org/wiki/Internet_of_things
[2] https://en.wikipedia.org/wiki/Smart_environment
[3] Malik, S.; Kim, D.H. A Study of Actuator Middleware in Indoor Actuator Networks. The 1st International Conference on Convergence Research Theory and Technology, CRTT 2017. Accepted, in press.
[4] Richardson L, Ruby S. RESTful web services. “O’Reilly Media, Inc."; 2008 Dec 17.
[5] Branch, J.W.; Davis, J.S.; Sow, D.M.; Bisdikian, C. In Sentire: A framework for building middleware for sensor and actuator networks, Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on, 2005; IEEE: pp 396-400.
[6] Ferreira, P.M.d.F.M.; Orvalho, J.o.; Boavida, F. In Middleware for embedded sensors and actuators in mobile pervasive augmented reality, INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, 2006; IEEE: pp 1-2.
[7] Erman, A.; van Hoesel, L.; Havinga, P. In Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with aerial objects, Proceedings of IEEE International Workshop on Safety, Security and Rescue Robotics, 2008; pp 1-6.
[8] Sommer, S.; Buckl, C.; Knoll, A. In developing service oriented sensor/actuator networks using a tailored middleware, Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on, 2009; IEEE: pp 1036-1041.
[9] Apel, S.; Bohm, K. In Towards the development of ubiquitous middleware product lines, International Workshop on Software Engineering and Middleware, 2004; Springer: pp 137-153.
[10] Al-Zoubi, K.; Wainer, G. Distributed simulation using restful interoperability simulation environment (rise) middleware. Intelligence-Based Systems Engineering, 129-157.
[11] Eisenhauer, M.; Rosengren, P.; Antolin, P. In A development platform for integrating wireless devices and sensors into ambient intelligence systems, Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops' 09. 6th Annual IEEE Communications Society Conference on, 2009; IEEE: pp 1-3.

Authors

Sehrish Malik, she received her B.S. in Computer Science from National University of Computer and Emerging Sciences (NUCES-FAST), Pakistan. In September 2014, she moved to Republic of Korea for M.S. studies and started working in the Protocol Engineering Laboratory (PEL), Sangmyung University (Cheonan-si Campus). After completing her M.S. in 2016, she moved to Jeju-do in September 2016, and started working as a Ph.D. research fellow in the Mobile Computing Laboratory (MCL), Jeju National University. Research interests include IoT, sensor networks, actuator networks, privacy, trust, and communication systems.

HeeDong Park, he received the B.S. degree in electronics engineering from the Kyungpook National University, Korea, in 1993, and the M.S. and Ph.D. degrees in information telecommunication the Kyungpook National University, Korea, in 1998 and 2005, respectively. Since 2007, he has been with the Korea Nazarene University, Republic of Korea, where he is currently a Professor of School of IT Convergence. His research interests include sensor networks, and mobile computing.

Do-Hyeun Kim, He received the B.S. degree in electronics engineering from the Kyungpook National University, Korea, in 1988, and the M.S. and Ph.D. degrees in information telecommunication the Kyungpook National University, Korea, in 1990 and 2000, respectively. He joined the Agency of Defense Development (ADD), from Match 1990 to April 1995. Since 2004, he has been with the Jeju National University, Republic of Korea, where he is currently a Professor of Department of Computer Engineering. From 2008 to 2009, he has been at the Queensland University of Technology, Australia, as a visiting researcher. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service and mobile computing.